MCMC in Baysian Variable Selection/Model Averaging

Merlise Clyde (Duke University)

11/14/2017

Example with Diabetes data

First lets load the data (from Hoff) and coerce them to be dataframes.

```
set.seed(8675309)
source("yX.diabetes.train.txt")
diabetes.train = as.data.frame(diabetes.train)
dim(diabetes.train)
```

```
## [1] 342  65
```

```
source("yX.diabetes.test.txt")
diabetes.test = as.data.frame(diabetes.test)
colnames(diabetes.test)[1] = "y"
```

Too many models 3.6893488×10^{19} to enumerate!

BMA using BAS

```
library(BAS)
diabetes.bas = bas.lm(y ~ ., data=diabetes.train,
                      method="MCMC", prior='hyper-g-n',
                      n.models = 10000,
                      MCMC.iteration=500000,
                      thin = 10, initprobs="eplogp")
```

- run a MCMC sampler that combines a Random Walk Metropolis algorithm with a random swap step.
- The algorithm stops when the number of iterations exceeds `MCMC.iterations` or `n.models` have been visited.
- `thin` save every 10th model.
- `initprobs` argument uses the p-values from OLS to compute an approximate Bayes Factor. These are used internally to sort the variables which provides improved memory usage.

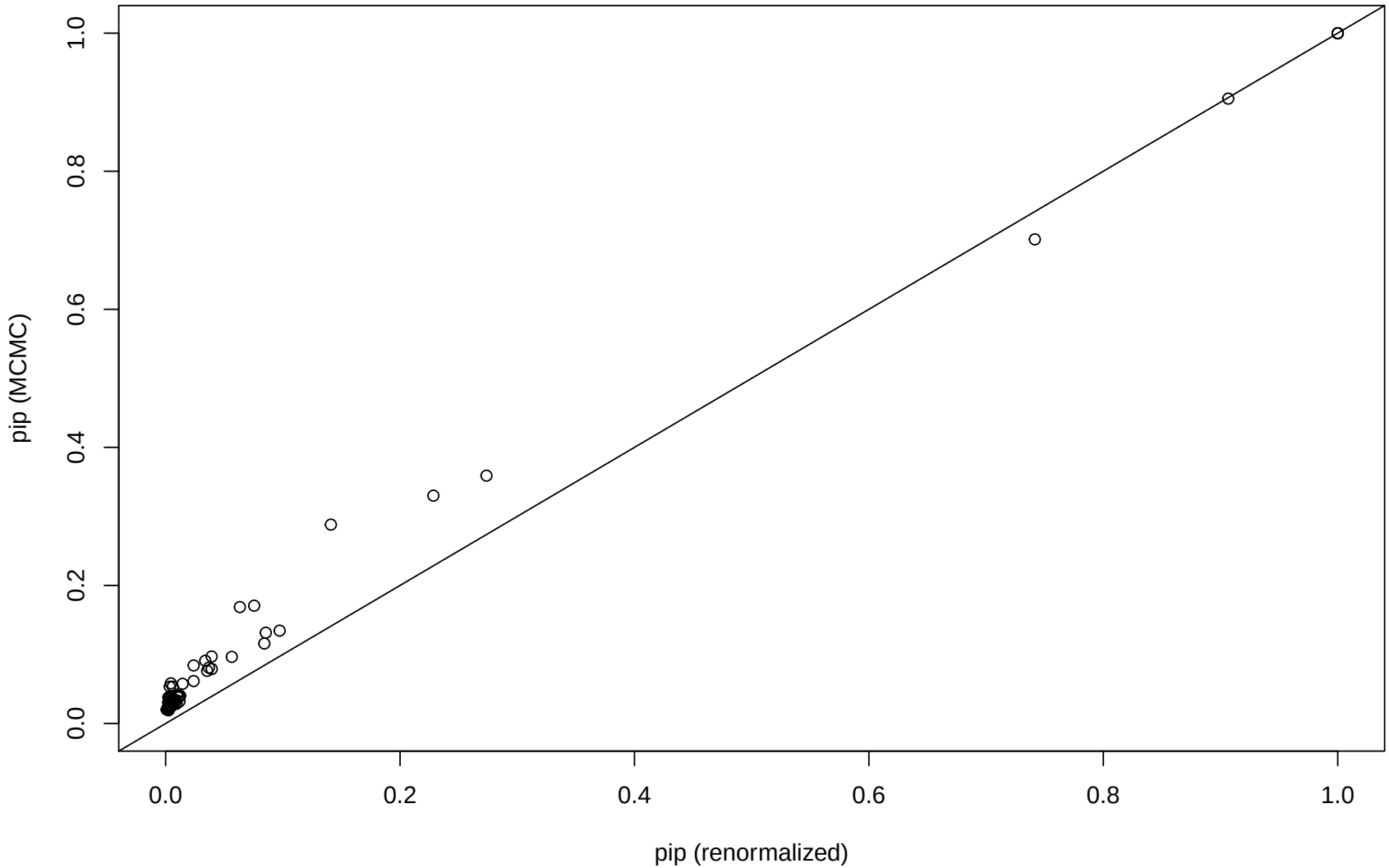
Checking Convergence

The function `diagnostics` compares

- estimates of posterior probabilities estimated from Monte Carlo frequencies (asymptotically unbiased) to
- estimates based on normalizing marginal likelihoods times prior probabilities of just the sampled models. (Fisher consistent; recovers exact under enumeration)
- should be equal if the chain has converged.

Convergence of Posterior Inclusion Probabilities

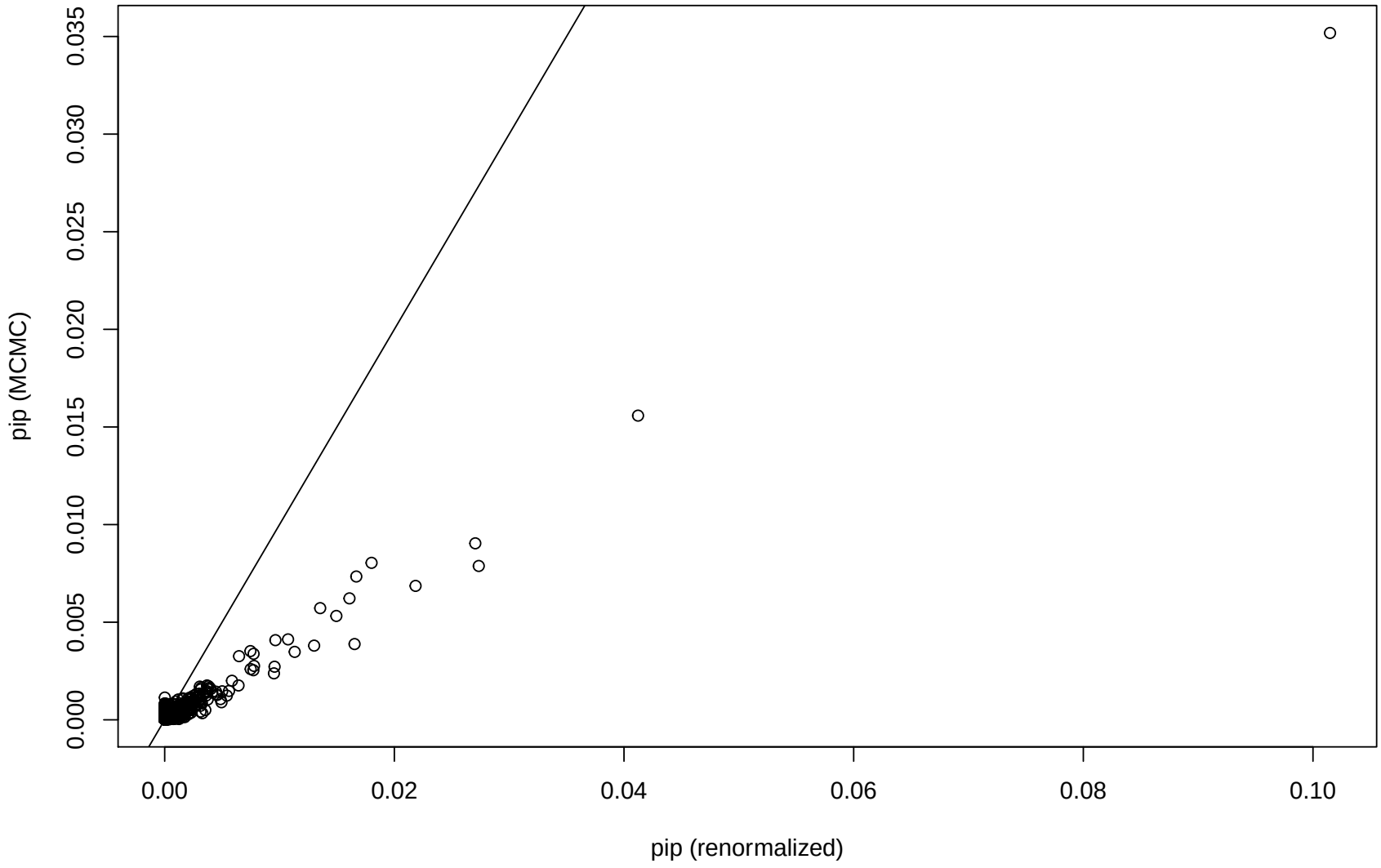
```
diagnostics(diabetes.bas, type="pip")
```



Close, but could run longer!

Convergence of model probabilities

```
diagnostics(diabetes.bas, type="model")
```



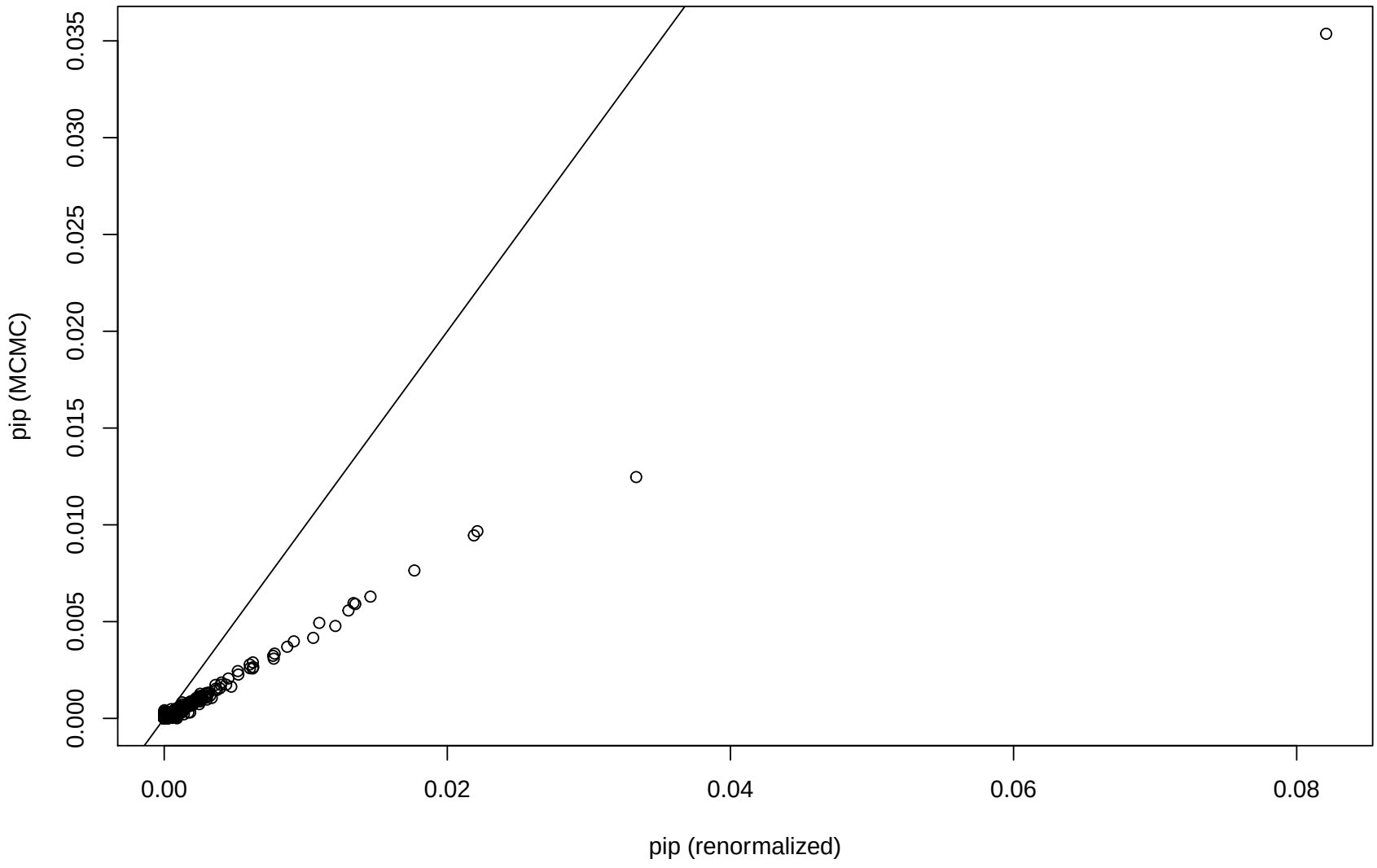
Clearly, not long enough.

rerun longer

```
diabetes.bas = bas.lm(y ~ ., data=diabetes.train,
                      method="MCMC", prior='hyper-g-n',
                      n.models = 150000,
                      MCMC.iterations=10^7,
                      thin = 64,
                      initprobs="eplogp")
```

Diagnostics

```
diagnostics(diabetes.bas, type="model")
```



Highest Probability Model

One problem with MCMC and model selection is that the "best" model may be visited very few times.

```
summary(diabetes.bas$freq)
```

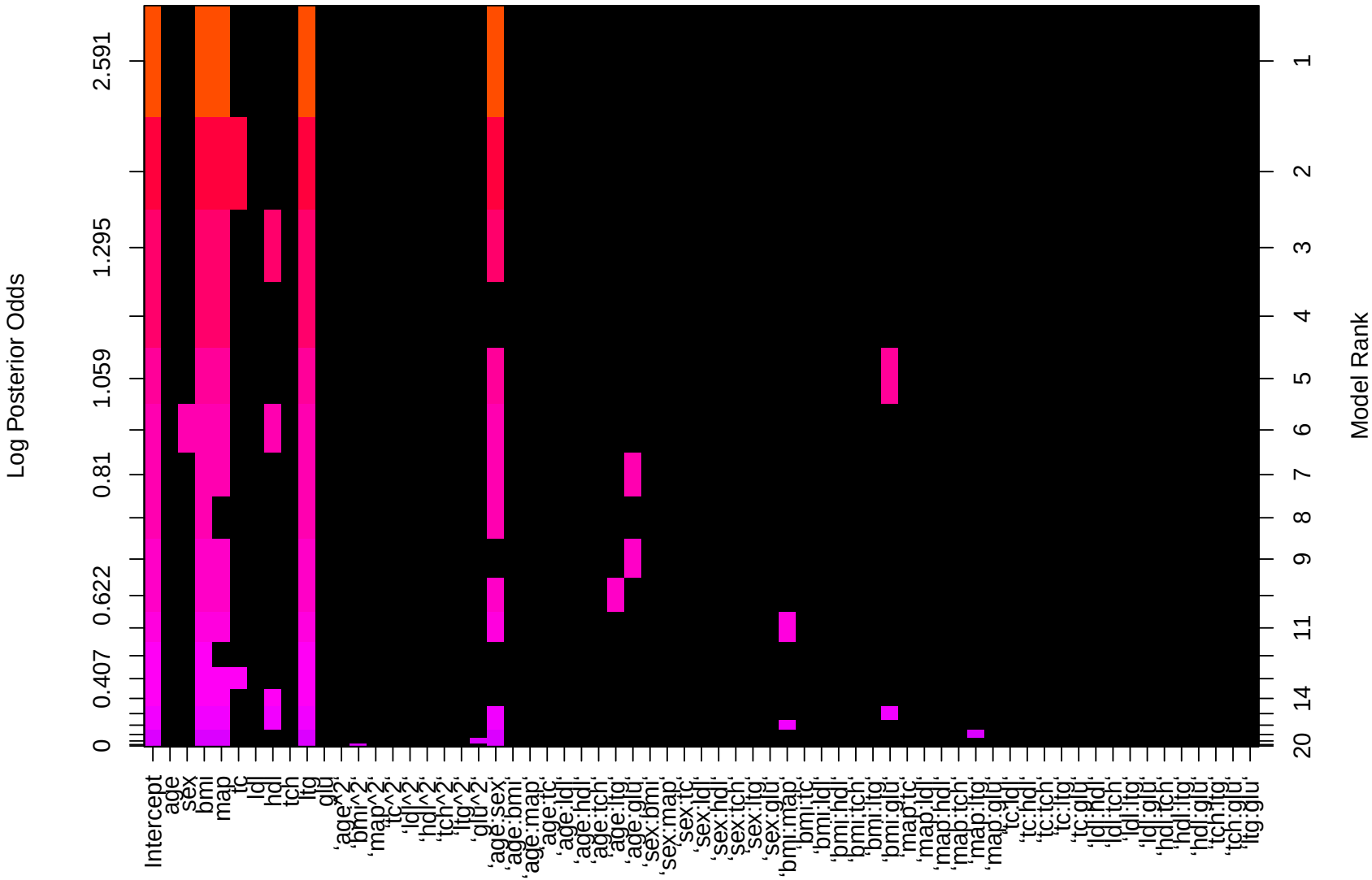
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.000    2.000    4.000    8.008    8.000 5526.000
```

Comments

- I routinely use much larger numbers of MCMC iterations than I see in papers if looking at BMA or model selection (i.e. millions rather than say 10,000)
- Explore different numbers of models - does it make a difference for estimating marginal inclusion probabilities or posterior inclusion probabilities? What about predictions under BMA?
- Check memory ! items can be large in R memory

What variables are included?

```
image(diabetes.bas)
```



Prediction

BMA for prediction and compute the MSE between the predicted and actual responses for the test data using BMA:

```
pred.bas = predict(diabetes.bas,
                   newdata=diabetes.test,
                   estimator="BMA", se.fit=T)
cv.summary.bas(pred.bas$fit, diabetes.test$y)
```

```
## [1] 0.6738473
```

Scoring

```
BAS::cv.summary.bas
```

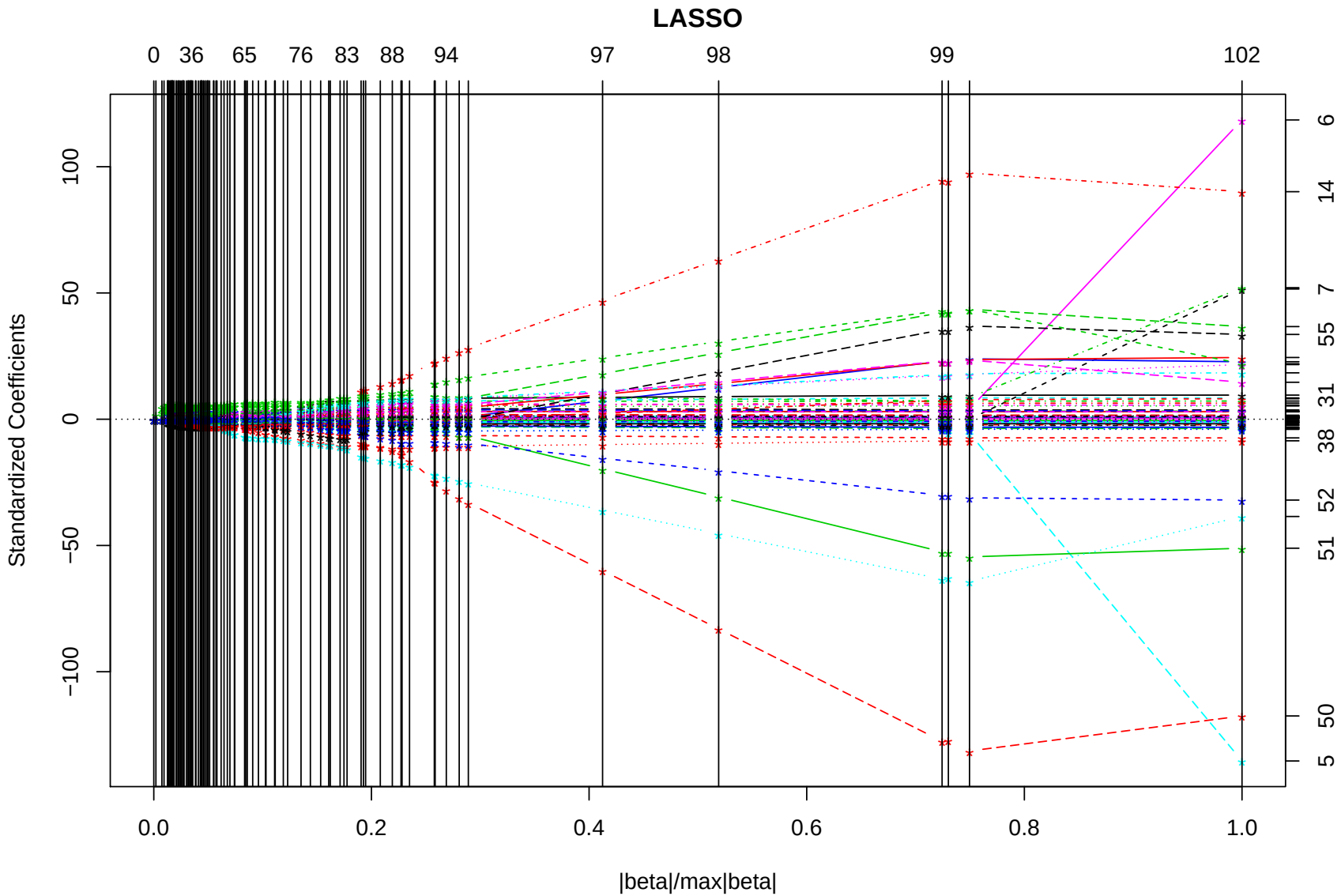
```
## function (pred, ytrue, score = "squared-error")
## {
##     if (length(pred) != length(ytrue)) {
##         warning("predicted values and observed values ar
##         return()
##     }
##     if (score == "miss-class") {
##         pred.class <- ifelse(pred < 0.5, 0, 1)
##         confusion = table(pred.class, ytrue)
##         error = (sum(confusion) - sum(diag(confusion)))/
##     }
##     else {
##         error = sqrt(sum((pred - ytrue)^2)/length(ytrue)
##     }
##     return(error)
## }
```

Fit the Lasso

```
suppressMessages(library(lars))
diabetes.lasso = lars(as.matrix(diabetes.train[, -1]),
                      diabetes.train[,1],
                      type="lasso")
```

Plot of Lasso

```
plot(diabetes.lasso)
```



Best Cp Model for Lasso

```
best = which.min(diabetes.lasso$Cp)
best
```

```
## 26
## 27
```

```
out.lasso = predict(diabetes.lasso, s=best[1],
                    as.matrix(diabetes.test[,-1]))
cv.summary.bas(out.lasso$fit, diabetes.test$y)
```

```
## [1] 0.6890253
```

Close

Choice of estimator

- BMA - optimal for squared error loss Bayes
- Highest Posterior Probability model (not optimal for prediction) but for selection
- Median Probabilty model (select model where PIP > 0.5) (optimal under certain conditions; nested models)
- Best Probability Model - Model closest to BMA under loss

HPM

```
pred.HPM = predict(diabetes.bas,
                   newdata=diabetes.test,
                   estimator="HPM", se.fit=TRUE)
cv.summary.bas(pred.HPM$fit, diabetes.test$y)
```

```
## [1] 0.6910106
```

MPM

```
pred.MPM = predict(diabetes.bas,
                   newdata=diabetes.test,
                   estimator="MPM")
cv.summary.bas(pred.MPM$fit, diabetes.test$y)
```

```
## [1] 0.6910106
```

BPM

```
pred.BPM = predict(diabetes.bas,
                   newdata=diabetes.test,
                   estimator="BPM")
cv.summary.bas(pred.BPM$fit, diabetes.test$y)
```

```
## [1] 0.6881558
```

Coverage

```
CI.bas = confint(pred.bas)
mean(CI.bas[,1] < diabetes.test$y & diabetes.test$y <= CI.b
```

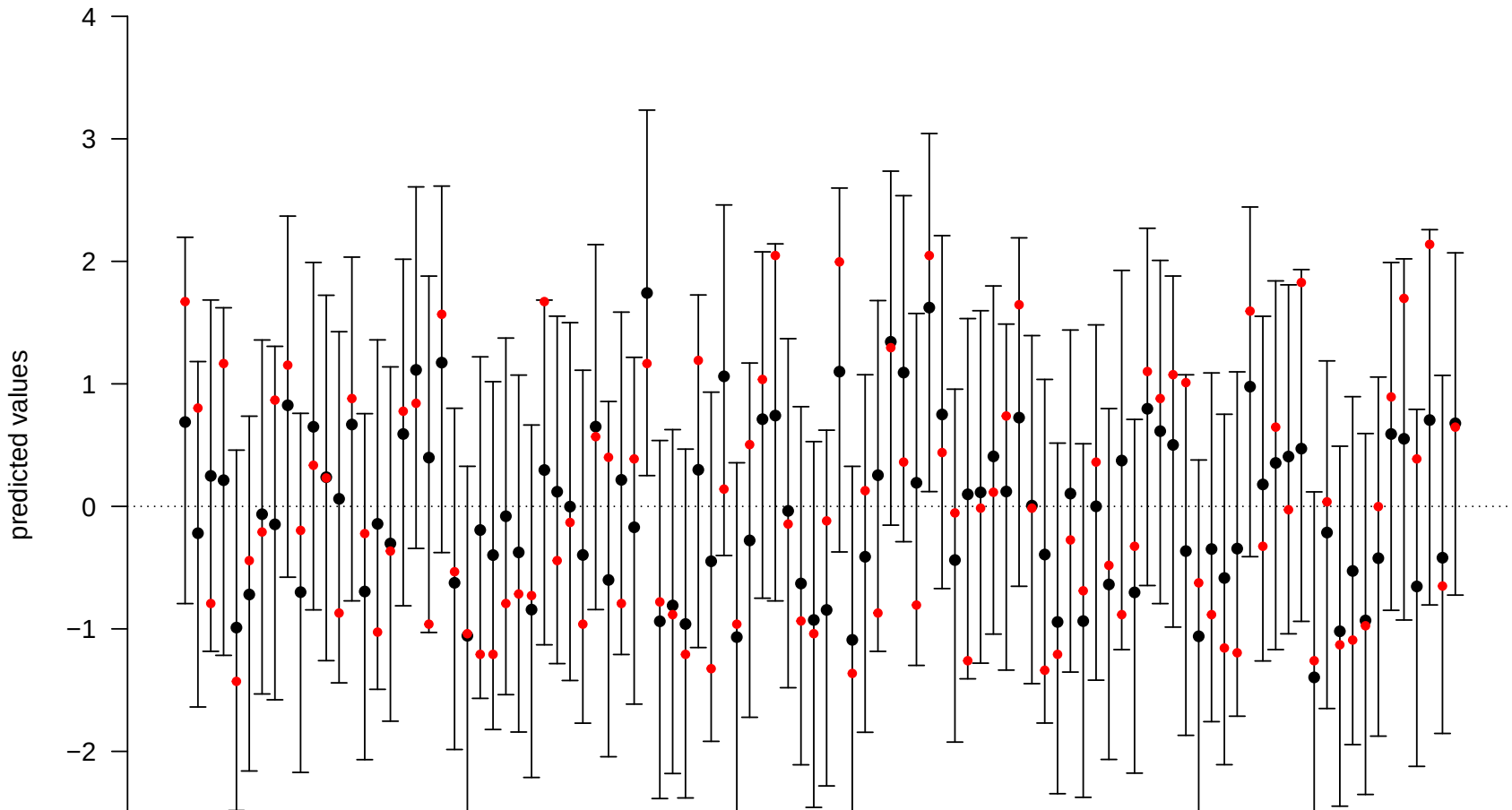
```
## [1] 1
```

Credible Intervals

```
plot(CI.bas)
```

```
## NULL
```

```
points(1:nrow(diabetes.test), diabetes.test$y, col="red", p
```



Coverage with HPM

```
CI.HPM = confint(pred.HPM)
mean(CI.HPM[,1] < diabetes.test$y & diabetes.test$y <= CI.H
```

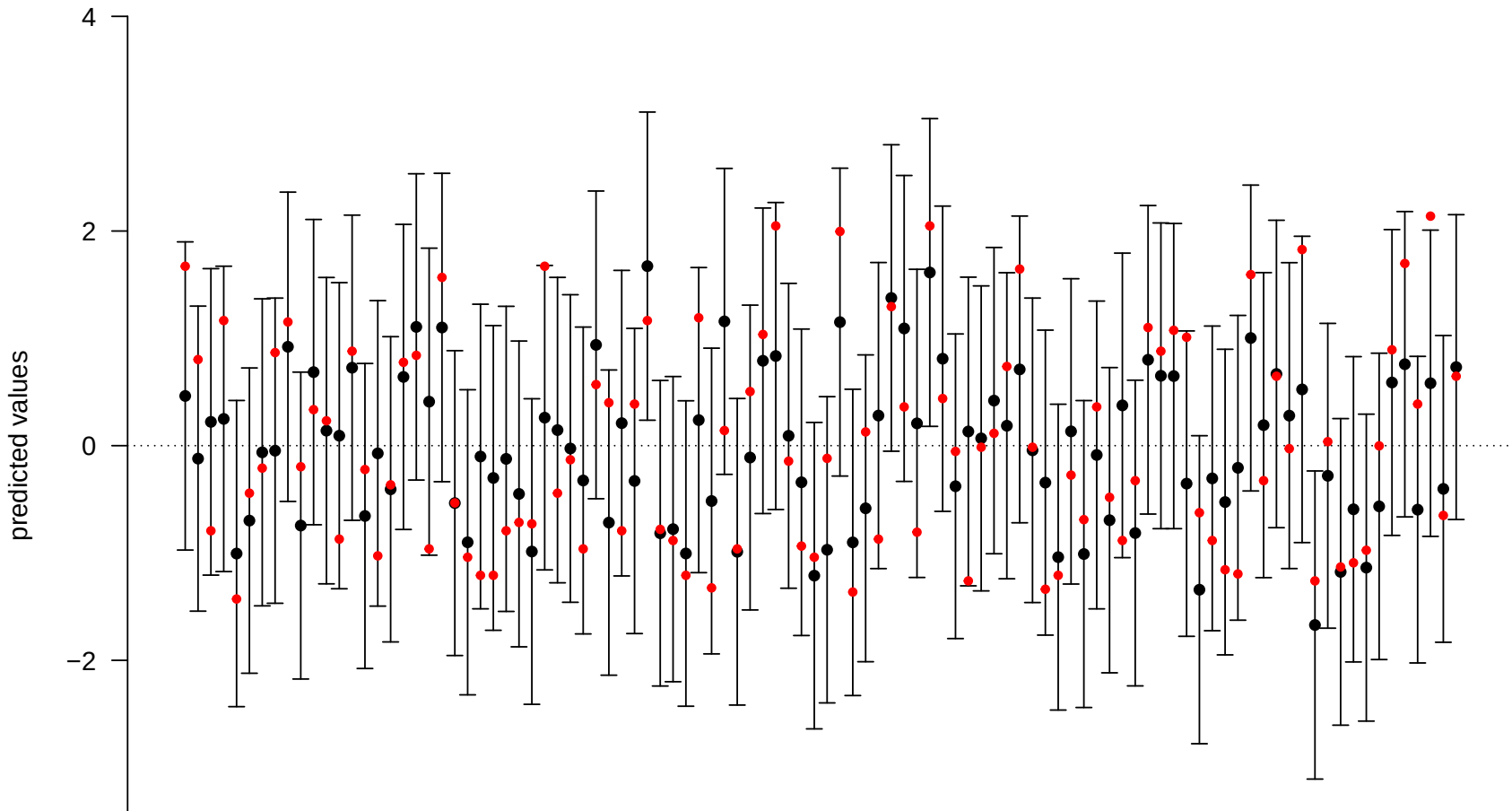
```
## [1] 0.99
```

Credible Intervals HPM

```
plot(CI.HPM)
```

```
## NULL
```

```
points(1:nrow(diabetes.test), diabetes.test$y, col="red", p
```



Calculation of CI

Summary

- prediction intervals account for model uncertainty
- multiple shrinkage with BMA
- choice of estimator and loss